Overview

This document covers the details of the benchmark applications associated with workstreams 7, 8, 9 and 10 (see section C.3.4.1 and other RFP sections for definitions related to the concept of “workstreams”).

A Microsoft Excel spreadsheet has been supplied for reporting initial results. Report all performance measures using only the benchmark supplied parallel infrastructure (if run in parallel). Report only actual measurements in the original table. Copy the table to report performance values obtained using alternate technologies; clearly mark projected values and provide details as to how the projection was derived.

Benchmark Components

Components for Workstreams 7, 8, 9 and 10

Summary:

The following is the application of the fundamental build/run design as outlined in the head benchmark instruction document to the content of workstreams 7, 8, 9 and 10:

1) Download bench.base.tgz. Unpack the benchmark build directory bench using gunzip and tar –xvf.

2) Change directory to the target benchmark bench/<model>/src

3) Edit the make to define for local conditions. You will probably have to modify the template to fit local conditions and current compiler, MPI library requirements, etc.

4) Make the executable; record the time make required to compile and link the executable in the spreadsheet provided..

5) Owing to their size, the benchmark run directories will be provided by tape. Offerors will supply a set of DLT format tapes to receive a copy of the benchmark run directories. Details for this process are pending and will be announced shortly.

The benchmark run directories associated with workstreams 7, 8 9 and 10 are:

a) RUC.tgz	(required - not available with first release)
b) ROMS.tgz	(required)
c) WRFchem_5KM.tgz	(required - not available with first release)
d) WRF+SI_5KM.tgz	(required)

Unpack the benchmark model run directory using gunzip and tar –xvf.

6) Copy the executable to the run directory.

7) Run the executable using appropriate MPI invocation; pipe stdout to a text file.

Of course any other means of capturing stdout is also acceptable. This copy of stdout and other ascii output files must be returned with the benchmark results for the timing and the reproducibility test described below.

8) Report the time in the spreadsheet provided. Report value rounded to the nearest second.

9) Report per process memory used.

10) Check results against verification files provided.

Model output files often have the same name regardless of PE count. Care must be taken not to accidentally overwrite output you wish to keep between consecutive model runs.

Note: The RFP benchmark will require a reproducibility test across PE count for some setting of the compiler, preferably that used for the performance benchmark itself. It is theoretically possible that high levels of compiler optimization might destroy reproducibility across PE count. To date, we have not seen this to be the case. Any mechanism causing loss of the capability to reproduce must be explained in detail. Use of such compiler optimizations will be weighed against the performance gain.

ROMS Benchmark code

There are two parts to the tests. The first part is a regression test which verifies the same answer is obtained for different processor layouts. The second part is a long run verifies that a long run on 32 processors with high optimization gives numbers that are reasonably close to the reference run which was made on ijet using the pg compiler with "-fast" optimization.

1) To compile the regression test:

cd to the subdirectory src.

There are three regression test makefiles. Two are for ijet (make.regress.pg and make.regress.ifc) and one is for jet (make.regress.jet). Modify one of these makefiles to produce a makefile called "Makefile" for your machine (see item #6 below) and then do:

```
gmake clobber
gmake
```

2) To run the regression test:

cd to the subdirectory Regression.

There are two versions of the regression test, "regress.ijet" for ijet and "regress.jet" for jet. Modify one of these scripts (see item #6 below) to create "regress" for your machine.

Submit the script "regress"

It should complete with the message : "Regression Successful"

3) To check your answers:

cd to the subdirectory NetCDFcompare.

In order to check your answers against the reference answers (produced on ijet with the pg compiler) you need a makefile to compile the NetCDF compare program. In the NetCDFcompare subdirectory there are three makefiles. Two are for ijet (Makefile.ijet.pg and Makefile.ijet.ifc) and one is for jet (Makefile.jet). Modify one of these makefiles to produce a makefile called "Makefile" for your machine.

The regression test should have produced three files: ocean_avg.nc, ocean_his.nc, and ocean_rst.nc. For each of these three files, do the following:

```
gen_compare.perl ../ijetRuns/ocean_avg.nc.regression
make
NetCDFcompare ../ijetRuns/ocean_avg.nc.regression ../Regression/ocean_avg.nc
```

If ocean_avg.nc.regression and ocean_avg.nc are suitably close, NetCDFcompare will print "passed". Otherwise, NetCDFcompare will print out the significant differences.

If you want NetCDFcompare to print out all the differences, go into the file stats.inc and set allowedMaxDiff to a negative number and recompile.

Note:

For the NetCDFcompare routine to work for very small and very large numbers, IEEE denormalization should be turned off when building NetCDF.

4) Longer run on 32 processors:

cd to the subdirectory src.

Optimization is turned off in the makefiles make.regress.*.
When doing long runs, recompile with optimization turned on (modify FFLAGS as in make.pg and make.jet)

cd to the subdirectory LongRun.

There is a script called qsubrun32.ijet for making a long run on 32p on ijet. Modify this script to create "qsubrun32" for your machine.

Submit the script "qsubrun32".

As did the script regress, qsubrun32 should produce three files; ocean_avg.nc, ocean_his.nc, and ocean_rst.nc. Again cd to the netCDFcompare subdirectory and, for all three files, recompile and run NetCDFcompare to check these files:

```
  gen_compare.perl ../ijetRuns/ocean_avg.nc.long
  make
  NetCDFcompare  ../ijetRuns/ocean_avg.nc.long ../LongRun/ocean_avg.nc
```

Included in the subdirectory ijetRuns is the ASCII output from the 32 processor run (qsubrun32.o1101306) as well as output from a 16 processor run (qsubrun16.o1119650) to show how roms scaled from 16p to 32p on ijet for this configuration.

5) Input parameters you may want to change:

For the long run, qsubrun32 uses run.in.
Parameters you may want to modify in run.in:

```
NTIMES - The number of time steps to run
NRST   - The frequency at which restart files are written
NHIS   - The frequency at which history files are written
NAVG   - The frequency at which average files are written
```

In run.in, these are all set to the same value so that exactly one of each file is written at the end of the run.

To change the process layout, modify NtileI and NtileJ in the grid.nl namelist file created in qsubrun32.

To change the grid resolution, modify Lm and Mm in grid.nl.

Currently, the length of the time step (DT) is small enough to handle a horizontal resolution of 2048x256 without violating the CFL condition (so it is smaller than it has to be for the regression test and the long run). Modify this parameter in run.in as appropriate.

6) Modifications for other machines:

The scripts make.regress.* set CFT to mpif90 and the Fortran compiler is determined by an environment variable. If your machine requires the compiler be named explicitly, modify CFT. This may require you to define MPI_INCDIR and include it in the compilation rule.

Modify FFLAGS as appropriate.

This code assumes a NetCDF library is available.

Modify NETCDF_INCDIR and NETCDF_LIBDIR as appropriate.

Modify CPP as appropriate.

Modify the batch header lines at the top of "regress" and "qsubrun32".

WRFV2 (Weather Research and Forecast) model.

Contents:

A) Directions for running a test case.
B) List of available test cases

(A) Directions for running a test case

A suite of tests for the WRF model ARW (Advanced Research WRF) core can be found in the directory "test". Each subdirectory in /test contains the necessary data (except for the real data case) and input files to run the test specific to that directory. To run specific test, build the WRF model and the necessary initialization routine by typing

```
-> compile "test_name"
```

in the top directory (the directory containing this README file).
For example, to build the executables for the 2D (x,z) squall line
example for Eulerian mass coordinate model, you would type the command
"compile em_squall2d_x".

after a successful build, go the the specific test directory:

```
-> cd test/"test_name"
```

run the initialization code

```
-> ideal.exe
```

and then run the simulation

```
-> wrf.exe
```

(B) Available Test Cases

The available test cases are

1) squall2d_x (test/em_squall2d_x)

   2D squall line (x,z) using Kessler microphysics
   and a fixed 300 m^2/s viscosity. The periodicity
   condition used in y causes the 3D model to produce a
   2D simulation. v velocity should be zero and there
   should be no variation in y in the results.

2) squall2d_y (test/em_squall2d_y)

   Same as squall2d_x, except with (x) rotated to (y).
   u velocity should be zero and there
   should be no variation in x in the results.

3) 3D quarter-circle shear supercell simulation
   (test/em_quarter_ss).

   Left and right moving supercells are produced.
   See the README.quarter_ss file in the test directory
   for more information.

4) 2D flow over a bell-shaped hill (x,z) (test/em_hill2d_x)

   10 km half-width, 2 km grid-length, 100 m high hill,
   10 m/s flow, N=0.01/s, 30 km high domain, 80 levels,
   open radiative boundaries, absorbing upper boundary.
   Case is in linear hydrostatic regime, so vertical tilted
   waves with ~6km vertical wavelength.

5) 3D baroclinic waves (test/em_b_wave)

   Baroclinically unstable jet u(y,z) on an
   f-plane. Symmetric north and south, periodic east and west
   boundaries. 100 km grid size 16 km top with 4 km damping layer.
   41x81 points in (x,y), 64 layers.

6) 2D gravity current (test/em_grav2d_x)

   Test case is described in Straka et al,
   INT J NUMER METH FL 17 (1): 1-22 JUL 15 1993.
   See the README.grav2d_x file in the test directory.